1. 协同程序：
   1. 基本概念：协程是一个分部执行，遇到条件（yield return 语句）会挂起，直到条件满足才会被唤醒继续执行后面的代码。简单来说：程序内部可中断，然后转而执行别的子程序，在适当的时候再返回来接着执行。
   2. 使用：
      1. 协程不是线程，也不是异步执行的。协程和 MonoBehaviour 的 Update函数一样也是在MainThread中执行的。协程可以和主程序并行运用，是一种对多线程的模拟机制。
      2. 用于延时一段时间后执行代码，或等其余某个操作执行完后再执行指定的代码。
2. ShaderLab：
3. 基本概念：Unity中编写Shader的语言称为ShaderLab,是Unity提供的一种编写语言，它以一种{}嵌套的形式来组织结构，并通过嵌入Cg/HLSL或GLSL程序片段的方式来实现着色器的编写。
4. Shader分类：在Unity中编写的Shader包括三类：
   1. Fixed Function Shader：也叫固定功能shader。不可编程，只能通过指令来修改状态或者属性。已经被Unity废弃，在发布时Fixed Fucntion Shader会自动转换Vertex and Fragment Shader。
   2. Surface-Shader：表面着色器。可编程，通过嵌入的Cg/HLSL片段来编写。编译指令#pragma surface指定。 Unity推荐使用的用于处理复杂光照的shader，它实际上是一种中间过程的Shader，是对Vertex and Fragment一种包装，简化了对光照处理相关的大量重复性指令，在发布时会自动转换成Vertex and Fragment Shader。
   3. Vertex and Fragment Shader：顶点-片段着色器。可编程，通过嵌入得Cg/HLSL片段来编写。通过编译指令#pragma vertex vert和 #pragma fragment frag来指定。示例中的就是这种类型。这是比较底层的着色器，可以自定义实现许多效果，但在实现复杂效果如复杂光照时需要手动处理相关的大量指令和过程。
5. 光照：
   1. 概述：Unity 中的光照可以粗略地分为实时和烘培，并且，两种光照可以组合使用，以创建沉浸式的场景光照。
   2. 实时光照：Unity 中的灯光——平行光、聚光灯和点光源，是实时的。这意味着，它们为场景提供直接光照，并且每桢更新。如果灯光和游戏对象在场景中移动，光照将立即更新。在场景视图和游戏视图中都可以观察这种变化。
   3. 烘培光照：Unity 可以计算复杂的静态光照效果（使用称为全局光照或 GI 的技术），并将它们存储在一张称为光照贴图的纹理文件中。这个计算过程成为烘培。
6. 帧动画：通过Unity自带动画编辑器，利用图片序列制作帧动画，并为动画剪辑、转换、交互提供方便的预览。使得动画可以更独立于程序和原型运行，在嵌入代码之前，就可以预览动画。
7. 图集：
   1. 概念：图集（Atlas）也称作 Sprite Sheet，是游戏开发中常见的一种美术资源。图集是通过专门的工具将多张图片合并成一张大图，并通过 plist 等格式的文件索引的资源。
   2. 用途：合成图集时会去除每张图片周围的空白区域，加上可以在整体上实施各种优化算法，合成图集后可以大大减少游戏包体和内存占用。多个 Sprite 如果渲染的是来自同一张图集的图片时，这些 Sprite 可以使用同一个渲染批次来处理，大大减少 CPU 的运算时间，提高运行效率。
8. Git版本控制：
   1. 概念：一个分布式版本控制软件。
   2. 主要功能：git是用于Linux内核开发的版本控制工具。它采用了分布式版本库的作法，不需要服务器端软件，就可以运作版本控制，使得源代码的发布和交流极其方便。
9. Adb：
   1. 概念：Android 调试桥 (adb) 是一个通用命令行工具，允许用户与模拟器或连接的 Android 设备进行通信。
   2. 用途：为各种设备操作提供安装和调试应用等功能的便利。